APPENDIX B

```
namespac  Syst  m.Storage
{

 // Executes a search across a specific type in an item context.
 public class ItemSearcher
 {

   Constructors

   public ItemSearcher();
   public ItemSearcher( Type targetType, ItemContext context );
   public ItemSearcher( Type targetType, ItemContext context,
                params SearchExpression[] filters );

   Properties

   // The filters used to identify matching objects.
   public SearchExpressionCollection Filters {get;}

   // The ItemContext that specifies the domains that will be searched.
   public ItemContext ItemContext {get; set;}

   // The search parameter collection.
   public ParameterCollection Parameters {get;}

   // The type the searcher will operate against. For simple searches this is the type of
   // the object that will be returned.
   public Type TargetType {get; set;}

   Search Methods

   // Find objects of TargetType that satisfiy the conditions specified by Filters. Returns
   // an empty FindResult if no such objects exist.
   public FindResult FindAll();
   public FindResult FindAll( FindOptions findOptions );
   public FindResult FindAll( params SortOption[] sortOptions );

   // Find any one object of TargetType that satisifies the conditions specified by Filters.
   // Returns null if no such object exists.
   public object FindOne();
   public object FindOne( FindOptions findOptions );
   public object FindOne( params SortOption[] sortOptions );

   // Find the object of TargetType that satisfies the conditions specified by Filters.
   // Throws ObjectNotFoundException if no such object was found. Throws MultipleObjects-
   // FoundException if more then one object was found.
   public object FindOnly();
   public object FindOnly( FindOptions findOptions );

   // Determine if an object of TargetType that satisfies the conditions specified by
   // Filters exists.
   public bool Exists();
```

```
// Creates an object that can be used to more efficiently execute the same search
// repeatedly.
public Prepar dFind Prepar Find();
public Prepar dFind PrepareFind( FindOptions findOpti ns );
public PreparedFind PrepareFind( params SortOption[] sortOptions );

// Retrieves the number of records that would be returned by FindAll().
public int GetCount();

// Asynchronous versions of various methods.
public IAsyncResult BeginFindAll( AsyncCallback callback,
                    object state );

public IAsyncResult BeginFindAll( FindOptions findOptions,
                    AsyncCallback callback,
                    object state );

public IAsyncResult BeginFindAll( SortOption[] sortOptions,
                    AsyncCallback callback,
                    object state );

public FindResult EndFindAll( IAsyncResult ar );

public IAsyncResult BeginFindOne( AsyncCallback callback,
                    object state );

public IAsyncResult BeginFindOne( FindOptions findOptions,
                    AsyncCallback callback,
                    object state );

public IAsyncResult BeginFindOne( SortOption[] sortOptions,
                    AsyncCallback callback,
                    object state );

public object EndFindOne( IAsyncResult asyncResult );

public IAsyncResult BeginFindOnly( AsyncCallback callback,
                     object state );

public IAsyncResult BeginFindOnly( FindOptions findOptions,
                     AsyncCallback callback,
                     object state );

public IAsyncResult BeginFindOnly( SortOption[] sortOptions,
                     AsyncCallback callback,
                     object state );

public object EndFindOnly( IAsyncResult asyncResult );

public IAsyncResult BeginGetCount( AsyncCallback callback,
                     object state );

public int EndGetCount( IAsyncResult asyncResult );
```

```
  public IAsyncR  sult BeginExists( AsyncCallback callback,
                         obj  ct stat   );

  public bool EndExists( IAsyncR  sult asyncResult );

// Options used when executing a search.
public class FindOptions
{

  public FindOptions();

  public FindOptions( params SortOption[] sortOptions );

  // Specifies if delay loadable fields should be delay loaded.
  public bool DelayLoad {get; set;}

  // The number of matches that are returned.
  public int MaxResults {get; set;}

  // A collection of sort options.
  public SortOptionCollection SortOptions {get;}

}

// Represents a parameter name and value.
public class Parameter
{
  // Initializes a Parameter object with a name and value.
  public Parameter( string name, object value );

  // The parameter's name.
  public string Name {get;}

  // The parameter's value.
  public object Value {get; set;}

}

// A collection of parameter name/value pairs.
public class ParameterCollection : ICollection
{

  public ParameterCollection();

  public int Count {get;}

  public object this[string name] {get; set;}

  public object SyncRoot {get;}

  public void Add( Parameter parameter );
  public Parameter Add( string name, object valu   );
  public bool Contains( Param  ter paramet  r );
  public bool C  ntains( string nam   );

  public void C  pyT  ( Param  ter[] array, int ind  x );
  v  id ICoII  cti  n.CopyT  ( Array array, int ind  x );

  IEnumerator IEnumerable.GetEnumerator();

  public void Remove( Parameter parameter );
  public void Remove( string name );

}

// Represents a search that has been optimized for repeated execution.
public class PreparedFind
{

  public ItemContext ItemContext {get;}

  public ParameterCollection Parameters {get;}

  public FindResult FindAll();

  public object FindOne();

  public object FindOnly();

  public bool Exists();

}

// Specifies sorting options used in a search.
public class SortOption
{

  // Initialize a object with default values.
  public SortOption();

  // Initializes a SortOptions object with SearchExpression, order.
  public SortOption( SearchExpression searchExpression, SortOrder order );

  // A search SearchExpression that identifies the property that will be sorted.
  public SearchExpression Expression {get; set;}

  // Specifies ascending or descending sort order.
  public SortOrder Order {get; set;}

}

// A collection of sort option objects.
public class SortOptionCollection : IList
{

  public SortOptionCollection();
```

```
    public S  rtOption this[int index] {g  t; set;}

    public int Add( S  rtOpti  n valu   );
    public int Add( S  archExpression   xpression, SortOrder   rd  r );
    int IList.Add( object value );

    public void Clear();

    public bool Contains( SortOption value );
    bool IList.Contains( object value );

    public void CopyTo( SortOption[] array, int index );
    void ICollection.CopyTo( Array array,  int index );

    public int Count {get;}

    IEnumerator IEnumerable.GetEnumerator();

    public void Insert( int index,  SortOption value );
    void IList.Insert( int index, object value );

    public int IndexOf( SortOption value );
    int IList.IndexOf( object value );

    public void Remove( SortOption value );
    void IList.Remove( object value );
    public void RemoveAt( int index );

    public object SyncRoot {get;}

  }

  // Specifies the sort order using in a SortOption object.
  public enum SortOrder
  {

    Ascending,
    Descending

  }
}
```